|  |
| --- |
| Augusto Xavier Novela                      **Impacto do Uso de um Sistema Informatizado de Gestão de Cutsheet – Caso de estudo: MOZTEX.S.A**  Licenciatura em Informática                        Universidade Pedagógica de Maputo  Maputo  2022 |
| Augusto Xavier Novela              **Impacto do Uso de um Sistema Informatizado de Gestão de Cutsheet – Caso**  **de estudo: MOZTEX.S.A**      Monografia apresentada ao Curso de Informática, Faculdade de Engenharia e Tecnologias, para a elaboração de Monografia.        Supervisor(a):  Mrs. Claúdia Jovo            Universidade Pedagógica de Maputo  Maputo  2022 |

# Índice

[Índice 1](#_Toc101256079)

[1.1. Introdução 4](#_Toc101256080)

[1.2. Problematização 5](#_Toc101256081)

[1.2.1. Hipóteses 8](#_Toc101256082)

[1.2.2. Motivação do Ploblema 8](#_Toc101256083)

[1.3. Objetivos 9](#_Toc101256084)

[1.3.1. Objetivo Geral 9](#_Toc101256085)

[1.3.2. Objetivos Específicos 9](#_Toc101256086)

[1.4. Metodologia de Pesquisa 9](#_Toc101256087)

[1.4.1. Abordagem (Tipo de pesquisa) 9](#_Toc101256088)

[1.5. Delimitação do Universo 9](#_Toc101256089)

[1.6. Técnicas de Recolha de Dados 10](#_Toc101256090)

[2.1. Fábrica 10](#_Toc101256091)

[2.1.1. Objectivo MOZTEX.SA 11](#_Toc101256092)

[2.2. Metodologia de desenvolvimento de Softwares 12](#_Toc101256093)

[2.2.1. Metodologias Tradicionais 12](#_Toc101256094)

[2.2.2. Metodologias Ágeis 12](#_Toc101256095)

[2.2.3. Diferença entre metodologias ágeis e tradicionais 13](#_Toc101256096)

[2.3. Extreme ProgrammingXP 14](#_Toc101256097)

[2.3.1. Valores do XP 14](#_Toc101256098)

[2.3.2. Práticas do XP 15](#_Toc101256099)

[2.3.3. Papeis dentro da Metodologia XP 16](#_Toc101256100)

[2.3.4. Fases do XP 16](#_Toc101256101)

[2.4. Linguagem de Modelação de Dados UML 18](#_Toc101256102)

[2.4.1. Diagramas Utilizados Pela UML 18](#_Toc101256103)

[2.4.2. Diagrama Caso de Uso 19](#_Toc101256104)

[2.4.3. Diagrama de Classes 19](#_Toc101256105)

[2.5. Sistema de Gestão de Base de Dados 20](#_Toc101256106)

[2.5.1. Base de Dados 20](#_Toc101256107)

[2.5.1.1. MYSQL 20](#_Toc101256108)

[2.6. Node.js 20](#_Toc101256109)

[2.6.1. Arquitetura do Node.js 21](#_Toc101256110)

[2.6.2. Características do Node.js 23](#_Toc101256111)

[2.7. Frameworks 23](#_Toc101256112)

[2.7.2. Vantagens dos frameworks 24](#_Toc101256113)

[2.7.3. Desvantagens dos frameworks 24](#_Toc101256114)

[2.8. Express.js 25](#_Toc101256115)

[2.8.1. Características do Express.js 25](#_Toc101256116)

[3. Referências bibliográficas 31](#_Toc101256117)

**Lista de Figura**

[Figura 1: Fases de desenvolvimento de software no modelo XP 17](#_Toc101253098)

[Figura 2: Processamento dos pedidos recebidos usando o loop do evento no node.js 21](#_Toc101253099)

[Figura 3: 1º Diagrama de Caso de Uso 28](#_Toc101253100)

[Figura 4: Diagrama de Caso de Uso 29](#_Toc101253101)

**Índice de Figuras**

[Tabela 1: Diferença entre metodologias ágeis e tradicionais 13](#_Toc101253157)

[Tabela 2: Requisitos funcionais do sistema de gestão de Cutsheet 26](#_Toc101253158)

[Tabela 4: Requisitos Não funcionais do sistema de gestão de Cutsheet 26](#_Toc101253159)

Devemos tomar cuidado para não fazer de nosso intelecto um deus. Ele tem, é claro,  
músculos poderosos, mas nenhuma personalidade.  
Ele não sabe governar, só servir.  
ALBERT EINSTEIN

CAPÍTULO I: INTRODUÇÃO

# Introdução

Graças aos grandes avanços da área de Tecnologia da informação (TI), nos dias que correm, a informação já não circula na sua maioria no formato físico, encontrando-se em muitos casos via Internet e outros meios eletrónicos. As empresas estão cada vez mais a aderir a circulação de informações no modo digital por meio de plataformas de gestão empresarial via web. E este facto deve-se pelas enumeras vantagens da utilização deste método, pois, este permite maior confidencialidade (às informações apenas estão disponíveis para quem está devidamente autorizado a acedê-las), Integridade (à informação permanece sem alterações não autorizadas) e por fim a sua disponibilidade (à informação esta sempre disponível para o utilizador no momento em que este precisa utiliza-la).

A gestão de cutsheets na MOZTEX.S.A é ainda feita de forma convencional, o que implica registar as encomendas e as requisições em papéis ou cadernos, deixando assim as informações susceptíveis a ameaças e/ou ataques.

Diante do exposto, remete-nos a definir estratégias que se circunscrevem na implementação de um sistema informatizado de gestão de cutsheets.

A presente pesquisa intitulada *Impacto do Uso de um Sistema Informatizado de Gestão de Cutsheet na MOZTEX.S.A*, visa analisar de forma aprofundada o funcionamento do sistema convencional em uso na fábrica.

# Problematização

A MOZTEX. S.A é uma empresa do ramo têxtil subsidiária da Aga Khan, localizada na Avenida Samora Machel, parcela de terreno 456, Município da Matola, Província de Maputo, tem como missão produzir peças de vestuário para crianças e adultos, mantas, uniformes escolares, fardamentos e outros. A empresa é constituída por 4 departamentos, nomeadamente Direção Geral, Administração, Recursos Humanos e Produção. O Departamento da administração é composto pelo sector de Contabilidade, secretaria e armazém, o departamento de Produção é composto pelo sector de corte, linha de produção e pelo acabamento.

A matéria-prima para efectivação do trabalho na Moztex. S.A é adquirido com recurso as encomendas feitas a companhia Sul Africana Kings Gate, que organiza as encomendas por *cutsheet,* este que é um documento que resume as informações e características de um produto, material ou equipamento. Cada material no *cutsheet* destina-se a uma respectiva produção, e nenhum material deve ser usado para uma produção que não seja à destinada. O documento de importação do material contem nome, preço, quantidade, cor, cutsheet para o qual será usado. Por sua vez Moztex.S.A recebe os produtos e faz a confirmação das quantidades e o seu estado antes de armazenar, caso verifique alguma anomalia no produto envia um *email* a reportar a situação, que pode ser de material estar em falta ou estar danificado. Após a confirmação, o material é organizado em prateleiras de acordo com o cutsheet, onde só será retirado mediante as requisições feitas pelo requerente (chamado na empresa de apontadora ou recolha) de cada sector ou linha de produção. Devendo está trazer na sua requisição, o material que necessita, a quantidade e o respetivo cutsheet, devidamente escritos no seu caderno de requisições e com assinatura do seu supervisor ou do coordenador da fábrica.

O armazém possui 4 tipos de cadernos de requisição, caderno de requisição de tecido– usado para a requisições feitas pelo sector do corte, onde deve-se registar o *bale* *number* (número de rolo), a metragem de cada rolo, a data e o cutsheet para o qual este saiu do armazém para no final do dia fazer-se cálculo total tirado para este uma vez que não pode exceder a metragem total do cutsheet**.** Caderno de requisição de acessórios **–** usadopelo setor de acabamentos, normalmente este requisita plásticos, cartolinas, ganchos (*jet clips* e *square clips*) entre outros materiais onde deve-se colocar a data, o material, quantidade e por fim a assinatura da recolha que vai levar o material, estas informações são registradas para controlar as quantidades exigidas por cada cutsheet para cada material não devendo exceder a quantidade exigida de acessórios**.** Caderno de requisição de calças **–** usado para registar as requisições do sector de produção (produção de calças), normalmente o material requisitado consistem em zippers, etiquetas, botões, papel do preço (tag), pega metálica calça (hook and bar) entre outros. Faz-se mediante ao registro da data, o cutsheet, o item, o tamanho para o caso de etiquetas e zippers, quantidade e o tipo de calça (school ou elástica), visando prevenir casos em que a recolha por seu descuido perde zipper ou etiqueta depois ter vindo requisitar no armazém uma vez que se tendo retirado o material do armazém este está na responsabilidade de quem o requisitou nesse caso a recolha ou apontadora. Caderno de requisição de camisas **–** usado para registar as requisições do sector de produção (produção de camisas) registrando a data, o cutsheet, o item, o tamanho e o tipo de calça (school manga curta, school longa ou fashion).

O processo do fabrico das peças de um cutsheet na MOZTEX.S.A obedece 4 fases nomeadamente corte, linha de produção, colagem e acabamento.

Tendo em conta que a gestão de *stock* obedece 3 grandes princípios: manutenção de *stock*, quanto e quando produzir com vista a garantir que não haja um rompimento de stock, denotando-se que na MOZTEX.S.A ainda vigora o uso de cadernos de anotações para gravar as informações referentes as requisições feitas pelos sectores acima citados, estes cadernos estão expostos a diversos factores que podem influenciar na perca de informação e denota-se também dificuldade de se obter dados estatístico do armazém de forma automática.

Diante do problema exposto surge a seguinte questão:

“ ***Até que ponto a implementação do Sistema de Gestão de Cutsheets pode melhorar o desempenho, eficiência e segurança dos inventários da MOZTEX.sa?***

# Justificativa

As aplicações webs permitem ampliar o universo de disseminação de informação a nível da sociedade. Elas têm tornado a informação acessível em diversos formatos e permitindo que aceso a elas seja possível independentemente da localização geográfica do usuário que quiser acede-la. A proposto deste estudo é buscar os dados necessários para fundamentar o desenvolvimento de uma plataforma web para auxiliar na gestão de stock da MOZTEX.S.A, para tal será analisado um conjunto de dados relacionados a gestão de stock no mundo em geral e em Moçambique em particular.

A escolha deste tema deveu-se ao facto de entender que o registro incorreto de dados referentes as encomendas e requisições efectuadas na fábrica em estudo, dificulta o acesso e interpretação da informação no momento de se fazer inventários.

No âmbito social é de extrema importância o desenvolvimento desta pesquisa, pôs a mesma tem como fim mostrar o impacto da implementação de sistema informatizado de gestão no sector fabril o que ainda não é muito acedido pelas pequenas médias e grandes fábricas nacionais.

No âmbito acadêmico este estudo contribuirá para alimentar futuras pesquisas com enfoque na gestão de *stock* uma vez que o contexto actual é caracterizado por situações que colocam em causa a segurança da informação.

No âmbito pessoal a escolha deste tema deve-se ao facto do mesmo estar ligado a área de actuação do pesquisador, o que irá contribuir no seu enriquecimento académico e desempenho profissional.

# Objetivos

## Objetivo Geral

* Conceber e Implementar uma plataforma para auxiliar na gestão do armazém da MOZTEX.S.A

### Objetivos Específicos

* Compreender os procedimentos usados na gestão do armazém da MOZTEX. S.A;
* Automatizar o processo de acesso aos dados do armazém da MOZTEX.SA;
* Propor a implementação de um sistema de gestão *stock* informatizado.

**Questão de Pesquisa**

* Quais são as limitações que a gestão de *stock* de forma convencional apresenta?
* Qual será o impacto da solução proposta em relação ao funcionamento da empresa?
* O sistema poderá facilitar as actividades dos funcionários no armazenamento de dados?
  + 1. Hipóteses

H1. Um sistema informatizado irá dinamizar os trabalhos da fábrica;

H2. Gestão de cutsheets de forma informatizada irá permitir mais independência por parte das pessoas responsáveis pelas requisições;

H3. Uma estrutura segura e modelada de gravação de dados irá permitir facilidade na interpretação de dados e na criação de relatórios.

# Metodologia de Pesquisa

## Abordagem (Tipo de pesquisa)

Segundo ANDER-EGG (1978, p. 28) pesquisa consiste em um “procedimento reflexivo sistemático, controlado e crítico, que permite descobrir novos factos ou dados, relações ou leis, em qualquer campo do conhecimento”.

A metodologia das pesquisas científicas pode ser classificada e definida conforme sua abordagem, finalidade e procedimentos técnicos empregados. (GIL, 2010 e VERGARA 2006)

A pesquisa é de caracter qualitativo que segundo (VIERA; ZOUAIN, 2006; BARDIN, 2011) visa principalmente, fazer análises qualitativas, caracterizando-se, em princípio, pela não utilização de instrumentos estatísticos na análise dos dados.

## Delimitação do Universo

Este estudo terá como universo os funcionários da MOZTEX.S.A, mas principalmente com os que estão diretamente ligados ao funcionamento do armazém (fiel de armazém, recolhas e gestor da fabrica) e pessoas com experiencia na área de fiéis de armazém.

## Técnicas de Recolha de Dados

Para a materialização da pesquisa recorreu-se a entrevista semi-estruturada, que segundo (GIL, 2008), é uma técnica em que o investigador se presente frente ao investigado e lhe formula perguntas, com o objetivo de obtenção dos dados que interessam à investigação. Essa entrevista consiste na coleta de dados nos intervenientes na gestão de cutsheets, nomeadamente a responsável pelo armazém, as recolhas ou funcionários que fazem as requisições e por fim as supervisoras de cada sector.

Recorreu-se também, revisões bibliográficas com vista a dar mais enfase e sustentar a pesquisa baseando-se material literário com conteúdo similar ou aproximado. (vide apêndice)

CAPÍTULO II: FUNDAMENTAÇÃO TEÓRICA

O presente capitulo incidir-se-á na definição de conceitos chaves para a compreensão do presente tema de pesquisa, a saber; fábrica, Metodologia de desenvolvimento de sofwares, engenharia de sofware, Linguagem de Modelação de Dados UML, sistemas de gestão de base de dados, frameworks

# 2.1. Fábrica

O dicionário define a fábrica como sendo Estabelecimento industrial onde se transformam matérias-primas em produtos destinados ao consumo, ou que se dedica à produção de outras mercadorias.

Já para a wekpedia, uma fábrica é um [edifício](https://pt.wikipedia.org/wiki/Edif%C3%ADcio) [industrial](https://pt.wikipedia.org/wiki/Ind%C3%BAstria) onde [trabalhadores](https://pt.wikipedia.org/wiki/Trabalhador) [manufaturam](https://pt.wikipedia.org/wiki/Manufatura) [bens](https://pt.wikipedia.org/wiki/Bens) ou supervisionam o funcionamento de [máquinas](https://pt.wikipedia.org/wiki/M%C3%A1quina) que [processam](https://pt.wikipedia.org/wiki/Processamento) um produto, transformando-o em outro. Possuem [armazéns](https://pt.wikipedia.org/wiki/Armaz%C3%A9m) e [depósitos](https://pt.wikipedia.org/wiki/Dep%C3%B3sito) com [equipamentos](https://pt.wikipedia.org/wiki/Equipamento) pesados, utilizados na produção da [linha de montagem](https://pt.wikipedia.org/wiki/Linha_de_montagem). [Arquetipicamente](https://pt.wikipedia.org/wiki/Arqu%C3%A9tipo), as fábricas acumulam e concentram recursos - [trabalhadores](https://pt.wikipedia.org/wiki/Trabalho_(economia)), [capital](https://pt.wikipedia.org/wiki/Capital_(economia)) e a [infraestrutura](https://pt.wikipedia.org/wiki/Infraestrutura) local.

# 2.2. Metodologia de desenvolvimento de Softwares

Segundo (INCRA, 2009) a metodologia de desenvolvimento de Softwares (MDS) tem como foco principal descrever e normalizar os processos de gerenciamento, de desenvolvimento e de manutenção de sistemas. E a aplicação de MDS visa promover o aumento na eficiência no gerenciamento e desenvolvimento de novos projetos ou manutenção de informação perspetiva.

O que vem ser enfatizado por (Moura, 2011) ao referir que a metodologia serve como um apoio de desenvolvimento, uma espécie de um manual de como fazer um software, contendo informações de como é feita a elaboração de requisitos, maior produtividade e redução de riscos.

## 2.2.1. Metodologias Tradicionais

As metodologias tradicionais são caracterizadas por valorizar o processo, a formalidade, a documentação e o planeamento inicial de um projeto. Qualquer membro do grupo pode ser facilmente substituído uma vez que tudo é muito bem estruturado antes do início de qualquer actividade. São exemplos desse tipo de metodologia o RUP e o Modelo em Cascata

O uso deste tipo de metodologia justifica-se em projetos de grande porte, onde a equipe de desenvolvimento não se encontra no mesmo espaço geográfico. Neste caso valoriza-se muito a documentação como forma de comunicação entre equipas (Cockburn, 2002).

## 2.2.2. Metodologias Ágeis

As metodologias ágeis são metodologias caracterizadas pelo facto do conhecimento não ficar todo armazenado na documentação, e sim na comunicação entre os membros envolvidos. A proposta é, por ser mais leve, proporcionar à equipe mais tempo para desenvolver, com menos interrupções. Podemos citar como exemplo das metodologias ágeis o SCRUM e o XP, este último que será usado para o desenvolvimento da aplicação em questão neste trabalho.

Normalmente criadas para que sejam incrementais e iterativas, e com o planeamento ocorrendo entre estas etapas, mudanças nos requisitos têm um impacto muito menor que nas suas contrapartes predeterminantes. O impacto de erros também é reduzido desta mesma forma. O desenvolvimento agora se torna algo dinâmico, maleável, rápido. (DOS SANTOS, 2007)

O principal problema deste tipo de metodologia é justamente depender demais da equipe. Algumas metodologias desta categoria exigem uma equipe mais madura de desenvolvedores para suceder. E, um membro da equipe passa a ser um pedaço do acervo da especificação do projeto, pois com ele fica o conhecimento sobre o mesmo. Conhecimento esse que nem sempre é documentado claramente no código-fonte. Esse tipo de metodologia, justamente por ser “centrada em pessoas” (Cockburn, 2002) torna os membros de um projeto muito menos substituíveis. Também é exigido um nível muito maior de disciplina e perícia, pois o desenvolvedor agora tem maior liberdade para usar as técnicas e ferramentas que desejar em certas etapas do processo. Cabe ao desenvolvedor também a especificação e planeamento durante uma etapa, iteração, incremento, *sprint*, ou o que for. (DOS SANTOS, 2007)

Nesses Como o *software* é construído pedaço por pedaço, o cliente tem agora a possibilidade de ir moldando o sistema a seu gosto, como um escultor de cerâmica, participando ativamente de todo o processo. Como se pôde notar, outro ponto crucial é a comunicação. Em metodologias deste tipo ela flui com muito mais facilidade, pois é feita normalmente face a face. Entretanto, este é o motivo principal para que o uso destas metodologias se restrinja a equipes pequenas. (DOS SANTOS, 2007)

## 2.2.3. Diferença entre metodologias ágeis e tradicionais

|  |  |  |
| --- | --- | --- |
|  | **Desenvolvimento tradicional** | **Desenvolvimento ágil** |
| **Principal suposição** | Sistemas completamente especificados | Melhorias contínuas |
| **Forma de gestão** | Comando e controle | Liderança e colaboração |
| **Comunicação** | Formal (comunicação baseada em artefactos) | Informal (comunicação oral ou  directa) |
| **Responsabilidade do Código** | Código tratado como subsistemas e com um membro responsável por cada. | Trata o código como coletivo |
| **Modelo de desenvolvimento** | Ciclo de vida (cascata e espiral) | Modelo de entrega  Evolucionária |
| **Controle de**  **Qualidade** | Muito planeamento. Teste tardio e complexo. | Controle de requerimento e testes contínuos |

Tabela 1: Diferença entre metodologias ágeis e tradicionais

Fonte: Autor

# 2.3. Extreme ProgrammingXP

Criado em 1997 por Kent Beck e Ward Cunningham é um conjunto de práticas, valores e princípios com foco em levar ao extremo boas práticas de programação e codificação visando prover maior qualidade do software, como também desenvolver software de forma leve (os processos são mantidos ao mínimo), humana (envolve pessoas como clientes, programadores entre outros) e disciplinada (há um conjunto de praticas e princípios que devem ser levadas e consideração para desenvolver a aplicação)

Para o XP os casos de teste nascem antes de se desenvolver a solução propriamente dita, porque segundo esta metodologia conhecendo os casos de teste é pouco provável que se desenvolva algo fora do que é esperado.

## 2.3.1. Valores do XP

1. **Comunicação –** para a construção de um bom projeto a comunicação é de extrema importância

e o XP trás práticas para que ela flua de forma rápida e compartilhada

1. **Criação de histórias do usuário** – criar uma história contendo as necessidades do usuário.
2. **Simplicidade** – a ideia é a solução não se basear em previsões mas sim no realmente se quer resolver agora de forma mais objetiva possível.
3. **Coragem** – quando se acha uma nova abordagem de desenvolvimento a equipe deve ser capaz de renunciar do projeto ou saber dizer não quando necessário e alterar o código quando for preciso.

## 2.3.2. Práticas do XP

1. **Cliente disponível** – é necessário que o cliente esteja disponível para resolver as dúvidas da equipe de desenvolvedores e de forma muito rápida.
2. **Planning game** – no jogo de planeamento são usadas as necessidades ou histórias das necessidades do cliente para se definir o cronograma, o levantamento de requisitos. Em geral são criadas regras para que o desenvolvimento seja rápido, eficiente (cumpra com o objetivo) e eficiente (cumpra com objetivo da melhor forma);
3. **Metáfora** – são usadas para facilitar a comunicação entre a equipe e consiste em atribuir nomes represantivos e comuns a equipe ao invés dos nomes próprios.
4. **Small releases** – são pequenas versões do sistema chamados por alguns autores de incrementos entregues ao cliente. Os incrementos são pequenas partes do sistema já em funcionamento. A ideia é o sistema ser utilizado pelo cliente antes mesmo de ser terminado permitindo assim o feedback do que já em funcionamento evitando assim erro das funcionalidades ainda em desenvolvimento
5. **Testes de aceitação –** definidos pelo usuário na fase inicial do projeto. São critérios de aceitação do sistema para satisfazer as necessidades do usuário.
6. **Criação de história de usuário** – criação de histórias contendo as necessidades do usuário.
7. **Test Driven Development (TDD)** – o desenvolvimento é feito orientado a testes
8. **Design simples** – as classes e métodos devem ser o mínimo possível oferecendo apenas aquilo que é necessário para atender os requisitos e não mais nada que aquilo.
9. **Integração continua** – cada função é testada de forma solitária para depois ser integrada as outras funcionalidades onde se fará mais um teste como um conjunto.
10. **Propriedade coletiva** – o código é uma propriedade coletiva onde todos são responsáveis por cada linha de código e usa-se as ferramentas de controlo de versões para manter as histórias pelas versões do código.
11. **Refactoracão** – programador pode modificar a estrutura do código sem alterar a arquitetura do sistema. Esta prática ocorre normalmente depois da integração das funcionalidades.
12. **Programação em duplas** – é das práticas mais famosas do XP, consiste em duas pessoas programar no mesmo computador comutando as tarefas entre programador e analista de código

## 2.3.3. Papeis dentro da Metodologia XP

1. **Programadores** – foco principal do XP sem hierarquia entre a equipe, ou seja, nenhum programador é melhor que o outro.
2. **Couch ou consultor** – é alguém que tem mais experiencia na utilização da metodologia XP.
3. **Tracker** – responsável por trazer informações para a equipe sobre o progresso para a tomada de decisões.
4. **Cliente** – deve estar sempre presente e pronto para responder as dúvidas dos programadores este deve fazer parte do processo.

## 2.3.4. Fases do XP

Segundo (VASCO,) o XP é composto por 6 fases, nomeadamente:

**Exploração**

Nesta fase o cliente escreve cartões de histórias, cada um contendo uma funcionalidade desejada para o primeiro *release*. Investigações são feitas para se certificar da fiabilidade e possíveis soluções para serem implementadas. Os programadores elaboram possíveis arquiteturas e tentam visualizar o futuro funcionamento do sistema considerando os mais diversos aspectos.

**Planeamento**

Nesta fase é acordado a data do lançamento do primeiro realese. Os programadores baseados nas histórias elaboradas pelo cliente identificam as dificuldades para cada um deles e baseados na sua velocidade de programação dizem quantas implementar em uma interação. Depois o cliente escolhe quais a história de maior valor para serem implementadas, esta actividade repete-se ate terminar as interações do realese o tempo para cada interação deve ser de uma semana e o tempo para cada realese deve ser de 2 a 3 meses. Em suma, nesta fase ocorre definição de prioridades entre as histórias junto com o cliente. Nesta etapa os programadores estimam o esforço e o cronograma para cada uma das estórias.

**Iterações para *Release***

É posto em prática o plano de interação em que os programadores seguem o plano de actividades onde faz-se os testes funcionais e unitários do projeto como também o refatoramento, codificação e realização do teste de aceitação e elas são escritas nas histórias do cliente.

**Validação para Produção (*Productionizing*)**: nessa fase são feitos testes extensivos e verificações para validação do software para ser utilizado em ambientes de produção. O cliente analisa o produto entregue e aprova ou desaprova, as informações dadas pelo cliente serão de extrema importância para as próximas versões.

**Manutenção**

Após o primeiro *release* para produção, há novas edições do sistema com novas funcionalidades.

Faz-se a manutenção propriamente dita, melhoramento e adaptação do código na medida que surgem novas aplicações ou pessoas

**Morte**

Esta fase referre-se ao fim do projeto atendendo as necessidades do cliente com um produto de qualidade ou inviabilidade económica ou alta presença de erros entre outros.

**Fases de desenvolvimento de software no modelo XP**

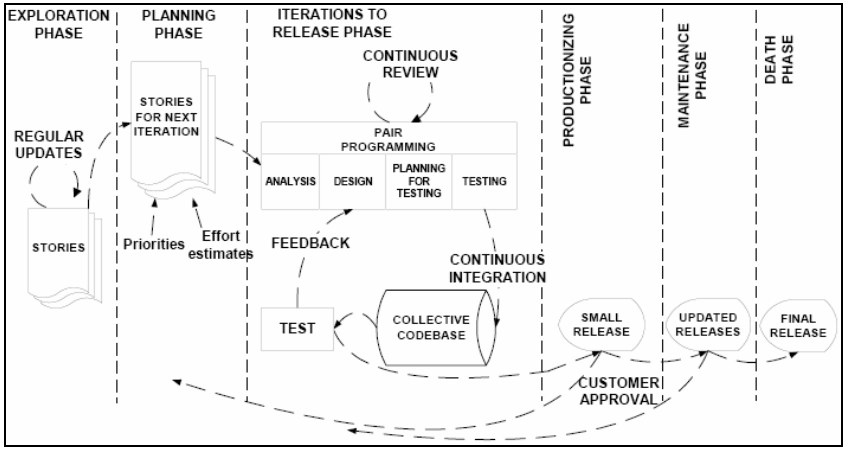


Figura 1: Fases de desenvolvimento de software no modelo XP

# Linguagem de Modelação de Dados UML

Para (GUEDES, 2011) A Unified Modeling Language ou simplesmente UML, é uma *linguagem de padronização* para a elaboração da estrutura de projetos de *software*, adequada para a modelagem de sistemas que vão desde aplicações web até sistemas complexos de tempo real.

O uso de qualquer ferramenta que suporte a UML permite analisar os requisitos futuros da aplicação e planejar uma solução que os contemple, representando os resultados através dos diagramas que a linguagem oferece.

## Diagramas Utilizados Pela UML

A UML baseia-se na utilização de nove diagramas diferentes, nomeadamente: diagrama de use case, de classes, de objeto, de estado, de sequência, de colaboração, de atividade, de componente e o de execução divididos em três modelos (estática, dinâmica e funcional). Modelagem estática é suportada pelo diagrama de classes e de objetos, que consiste nas classes e seus relacionamentos. Os relacionamentos podem ser de associações, herança (generalização), dependência ou refinamentos. Enquanto, que a modelagem dinâmica é suportados pelos diagramas de estado, sequência, colaboração e atividade. E por fim a modelagem funcional é suportado pelos diagramas de componente e execução.

Neste contexto, o autor abordara apenas dois destes diagramas: diagrama de caso de uso e diagrama de classes.

## Diagrama Caso de Uso

O diagrama caso de uso é uma técnica usada para descrever e definir os requisitos funcionais de um sistema. Eles são escritos em termos de atores externos, caso de uso e o sistema modelado. Os atores representam o papel de uma entidade externa ao sistema como um usuário, um hardware, ou outro sistema que interage com o sistema modelado. Os atores iniciam a comunicação com o sistema através dos casos de uso, onde o caso de uso representa uma sequência de ações executadas pelo sistema e recebe do ator que lhe utiliza dados tangíveis de um tipo ou formato já conhecido, e o valor de resposta da execução de um caso de uso também já é de um tipo conhecido, tudo isso é definido juntamente com o caso de uso através de texto de documentação.

## Diagrama de Classes

Este tipo de diagrama visa demonstrar a estrutura estática das classes de um sistema onde estas representam as tudo que é gerenciadas pela aplicação modelada. Classes podem se relacionar com outras através de diversas seguintes formas: associação (conectadas entre si), dependência (uma classe depende ou usa outra classe), especialização (uma classe é uma especialização de outra classe), ou em pacotes (classes agrupadas por características similares). Todos estes relacionamentos são mostrados no diagrama de classes juntamente com as suas estruturas internas, que são os atributos e operações. O diagrama de classes é considerado estático já que a estrutura descrita é sempre válida em qualquer ponto do ciclo de vida do sistema. Um sistema normalmente possui alguns diagramas de classes, já que não são todas as classes que estão inseridas em um único diagrama e uma certas classes pode participar de vários diagramas de classes.

# Sistema de Gestão de Base de Dados

Um Sistema de Gestão de Bases de Dados ou simplesmente SGBD é o software que gere o armazenamento, manipulação e pesquisa dos dados existentes na base de dados, funcionando como um interface entre as aplicações e os dados necessários para a execução dessas aplicações. (LUCAS, 2008) O principal objetivo dos sistemas de gestão de base de dados é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, a persistência, a manipulação e a organização dos dados. São exemplos: IMS, DB2, Ingres, Informix Dynamic Server, Oracle Server, Sybase SQL Server, Microsoft SQL Server, Access.

### ****Base de Dados****

Podemos dizer, assim, que uma base de dados é um conjunto organizado de dados (segundo um modelo de dados) existentes num sistema informático disponível a todos os utilizadores ou processamentos da organização. O acesso e utilização a uma base de dados é feita através de um software específico chamado de sistema de gestão de base de dados.

### MYSQL

MySQL é um sistema de gerenciamento de banco de dados caracterizado pela utiliza a linguagem SQL como interface. Lançado sobre a licença GPL, atualmente é mantida pela Oracle Corporation. É multiplataforma, ou seja, possui suporte para diferentes sistemas operacionais, entre eles, o Windows, Linux e Mac.

O MYSQL é gratuito; Open source; Possui uma comunidade ativa; Pode ser utilizado em qualquer tipo de aplicação (das mais simples às mais robustas); Facilidade de programação e aprendizado.

# [Node.js](https://www.opus-software.com.br/node-js/)

A wikipedia define o node.js como “uma compilação empacotada do motor JavaScript V8 (é o motor de execução JavaScript) do Google, a camada de abstração da plataforma libuv, e uma biblioteca central, que por si só é escrita principalmente em JavaScript”. A versão atual do Long Term Support (LTS) é o Node v8.9.3, enquanto a versão mais recente é o Node 9.

Node foi projetado para otimizar a taxa de transferência e a escalabilidade em aplicações web. É uma ótima combinação para resolver muitos problemas comuns no desenvolvimento da web (por exemplo, aplicações em tempo real). O código é escrito em "JavaScript simples e antigo". Isso significa menos tempo gasto para lidar com mudanças de código entre navegador e servidor web, não sendo necessária uma mudança na linguagem.

O Gerenciador de Pacotes do Node (NPM, na sigla em inglês) provê acesso a centenas de milhares de pacotes reutilizáveis. NPM possui a melhor coleção de dependências e também pode ser usado para automatizar a maior parte da cadeia de ferramentas de compilação.

É portátil, com versões para diferentes sistemas operacionais, como Microsoft Windows, OS X, Linux, Solaris, FreeBSD, OpenBSD, WebOS e NonStop. Além disso, tem excelente suporte de muitos provedores de hospedagem na web, que muitas vezes fornecem documentação e infraestrutura específica para hospedar sites desenvolvidos em Node.

## Arquitetura do Node.js

O Node.js usa a arquitetura “Single Threaded Event Loop” para lidar com vários clientes ao mesmo tempo.

Em um modelo de resposta a solicitações multi-threaded (usado por diversos servidores webs de linguagens tradicionais), vários clientes enviam uma solicitação, e o servidor processa cada um deles antes de enviar a resposta de volta. No entanto, múltiplas threads são usados para processar chamadas simultâneas. Estes tópicos são definidos em um pool de tópicos, e cada vez que uma solicitação chega, um tópico individual é designado para tratá-la.

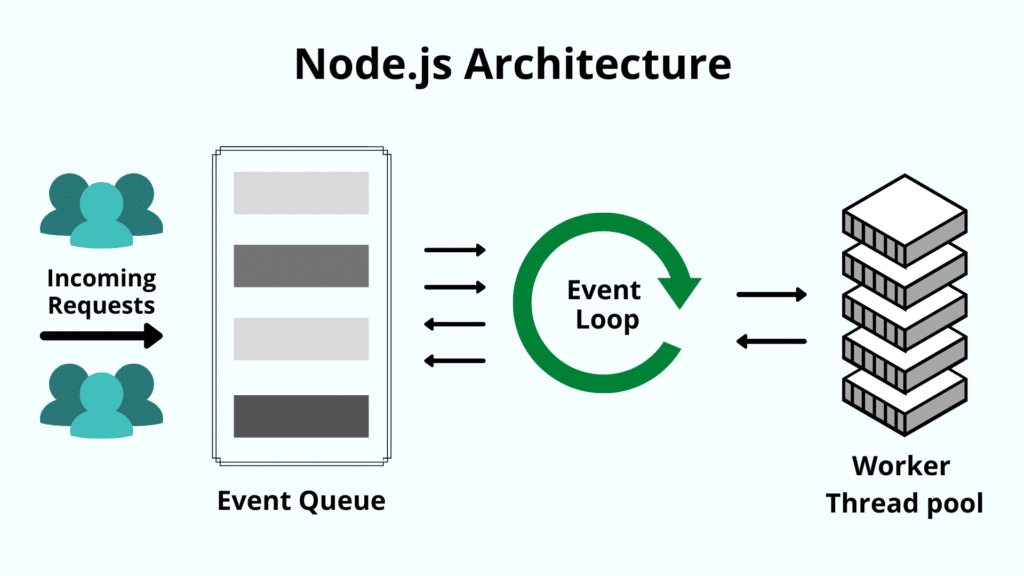
[](https://kinsta.com/wp-content/uploads/2021/03/Nodejs-Architecture.png)

Figura 2: Processamento dos pedidos recebidos usando o loop do evento no node.js

Fonte: Autor

 O Node.js funciona de forma diferente. Vamos dar uma olhada em cada passo que ele passa:

1. O Node.js mantém um pool limitado de threads para atender aos pedidos.
2. Sempre que um pedido chega, o Node.js o coloca em uma fila.
3. Agora, o “loop de eventos” – o componente central – é o que está em jogo. Este loop de eventos espera por solicitações indefinidamente.
4. Quando um pedido chega, o laço a pega da fila e verifica se requer uma operação de entrada/saída (E/S) de bloqueio. Caso contrário, ele processa a solicitação e envia uma resposta.
5. Se a solicitação tiver uma operação de bloqueio a realizar, o loop do evento atribui um fio do pool interno de threads para processar a solicitação. Há poucas roscas internas disponíveis. Este grupo de roscas auxiliares é chamado de grupo de trabalhadores.
6. O loop do evento rastreia os pedidos de bloqueio e os coloca na fila uma vez que a tarefa de bloqueio é processada. É assim que mantém sua natureza de não-bloqueio.

Como o Node.js utiliza menos threads, ele utiliza menos recursos/memória, resultando em uma execução mais rápida das tarefas. Portanto, para nossos propósitos, esta arquitetura de rosca única é equivalente à arquitetura multi-tarefa. Quando se precisa processar tarefas com uso intensivo de dados, então o uso de linguagens multi-tarefa como Java faz muito mais sentido. Mas para aplicativos em tempo real, o Node.js é a escolha óbvia.

## Características do Node.js

O Node.js tem crescido rapidamente nos últimos anos. Isto se deve à vasta lista de características que ele oferece:

1. **Fácil**  – Node.js é bastante fácil de começar. É a escolha ideal para iniciantes no desenvolvimento web. Com muitos tutoriais e uma grande comunidade iniciada, é muito fácil.
2. **Escalável** – Proporciona uma vasta escalabilidade para aplicativos. O Node.js, sendo de rosca única, é capaz de lidar com um enorme número de conexões simultâneas com alto rendimento.
3. **Veloz** – A execução da linha Non-blocking torna o Node.js ainda mais rápido e mais eficiente.
4. **Pacotes** – Um vasto conjunto de pacotes open-source Node.js está disponível que pode simplificar seu trabalho. Há mais de um milhão de pacotes no ecossistema NPM atualmente.
5. **Forte back-end** – Node.js forte é escrito em C e C++, o que o torna mais rápido e acrescenta recursos como suporte de rede.
6. **Multiplataforma** – plataforma permite criar sites SaaS, aplicativos desktop e até mesmo aplicativos móveis, todos usando o Node.js.
7. **Manutenível**  – Node.js é uma escolha fácil para os desenvolvedores, já que tanto o frontend quanto o backend podem ser gerenciados com JavaScript como uma única linguagem.

# Frameworks

Segundo Mattsson (1996, 2000), um framework é uma arquitetura desenvolvida com o objetivo de atingir a máxima reutilização, representada como um conjunto de classes abstratas e concretas, com grande potencial de especialização.

Já segundo Buschmann et al. (1996), Pree (1995) e Pinto (2000) um framework é definido como um software parcialmente completo projetado para ser instanciado.

Os Frameworks (termo inglês para estrutura) são estruturas de códigos genérica que têm como objetivo prover uma nova função dentro do seu código, provendo funções de ajuda e tomando decisões que facilitem o trabalho do desenvolvedor. O seu funcionamento é equiparado ao jogo de quebra cabeça onde cada peça é capaz de se encaixar nos mais diversos lugares e conectar todos as linhas de código de uma maneira quase perfeita.

No ambiente web eles oferecem soluções para problemas como estratégias de roteamento, compilação, gerenciamento de estados, além de estruturar e gerar arquivos e configurações do projeto.

Possui como principal objetivo resolver problemas recorrentes com uma abordagem mais genérica. Ele permite ao desenvolvedor focar nos “problemas” da aplicação, não na arquitetura e configurações.

* + 1. **Funcionamento dos Framework**

A construção dos frameworks é baseada em várias bibliotecas de códigos já prontas espalhadas no mundo. Qualquer um com vontade e conhecimento suficiente para tal pode construir e disponibilizar um framework interessante. Devida a essa facilidade no compartilhamento, esse conjunto de códigos funcionam de maneira ampla entre as linhas de programação para garantir seu bom funcionamento. E a maioria dos frameworks usam os princípios de orientação do objeto com herança, polimorfismo e abstração.

### Vantagens dos frameworks

* Incrível capacidade de adaptação a diversos contextos.
* Legibilidade
* Maior da produtividade
* Códigos sem erros
* Maior segurança
* Projeto padronizado
* Redução de custos e tempo
  + 1. **Desvantagens dos frameworks**
* Dificuldade de sua configuração
* A dependência
* Mais códigos que o necessário
* Causa dependência

# 2.8. Express.js

O Express.js é um Framework para o desenvolvimento de aplicações JavaScript com o Node.js. E a sua versão inicial foi lançada no ano de 2010. Trata-se de um framework open source, sobre a licença MIT que já foi desenvolvido para otimizar a construção de aplicações web e APIs, tornando-se um dos Frameworks mais populares da internet e que utiliza o Node para execução do javascript como linguagem de back-end.

## 2.8.1. Características do Express.js

Dentre suas principais características, podemos citar: o facto de ser um sistema de rotas completas; Possibilita o tratamento de exceções dentro da aplicação; Permite a integração de vários sistemas de templates que facilitam a criação de páginas web para suas aplicações; Gerência diferentes requisições HTTP com seus mais diversos verbos;

**CAPÍTULO III: APRESENTAÇÃO E DISCUSSÃO DOS RESULTADOS**

Neste capítulo demostrar-se-á a análise do conteúdo para resolver o problema, as técnicas  
usadas para recolha de requisitos, o uso do método e o uso da metodologia e os testes.

**3.1. Exploração**

A iniciação é a primeira fase das seis (6) fases da metodologia XP e consiste em ter o primeiro contacto com o cliente, ouvir as suas histórias (Funcionamento do sistema) fazer a recolha de dados e obter a ideia geral de como será feito o sistema. A recolha das histórias foi feita na zona d

**3.1.1. 1ª Exploração**

Na primeira exploração foi feita a apresentação do sistema convencional em uso na MOZTEX.SA. Essa reunião foi realizada pelo autor junto com o fiel de armazém Juaneta Simba e o coordenador da fabrica Mr Sami Akur, onde foi explicado o funcionamento geral do sistema, suas funções, limitações e etc. Seguidamente foi feita a recolha de dados na base de  
entrevistas, inquérito e por questionário, onde foi entrevistado um grupo de funcionários (Pointers) responsável por fazer as requisições no armazém e estas responderam certas questões sobre o seu funcionamento e melhorias que acham que deviam ser apontadas. Em simultâneo foi dado um formulário sobre aspectos específicos do actual funcionamento. Esse processo levou aproximadamente um mês devido a pandemia da corona vírus era difícil se encontrar com a responsável do armazém. De seguida foi a vez de se avaliar os resultados obtidos dos formulários, de modo a  
saber se é viável ou não a criação de um sistema informatizado de gestão de cutsheets.

**3.1.2. 2ª Exploração**

Desta, foi a vez de se apresentar a proposta de implementação de um sistema de gestão de cutsheets ao Kenny (Contabilista) e ao Fazimil Faizel (Director Geral da Fabrica). A reunião consistiu na demostração da viabilidade e necessidade de implementação de o sistema informatizado no armazém da fábrica. Também foram dadas certas sugestões por parte do director: principalmente que dados e relatórios que deviam ser gerados pelo sistema.

**3.2. 3ª Exploração**

Depois de se ouvir as histórias do cliente repetidas vezes, fez-se uma tabela com as necessidades principais (primeiros requisitos funcionais) por parte do autor e foi se ter com o cliente para comprovar se é exatamente o que ele quer numa fase inicial, pois, o XP permite desenvolver o sistema em partes. De igual forma, foram produzidos os primeiros diagramas de classe e de caso de uso.

**3.2.1. Requisitos Funcionais**

Tabela 2: Requisitos funcionais do sistema de gestão de Cutsheet

|  |  |  |  |
| --- | --- | --- | --- |
| ***Nº*** | **Requisito** | **Descrição** | **Nome do Caso de Uso** |
| **RF01** | Permitir o registro do cutsheet | O sistema deve permitir o cadastro dos cutsheets. | Registar cutsheet |
| **RF02** | Permitir o registro do Material | O sistema deve permitir o cadastro dos Material. | Registar Material |
| **RF03** | Permitir o registro de Material danificados | O sistema deve permitir se registra todos Material que não estiverem em condições de serem usados | Registar Material Danificado |
| **RF04** | Permitir efetuar Requisição | O sistema deve permitir com que o administrador ou outro usuário com as credencias autorizadas efetua a Requisição dos Material. | Efectuar Requisição |
| **RF05** | Permitir o registro do Stock | O sistema deve permitir registra Stock | Cadastrar Stock |
| **RF06** | Permitir o registro da Operário | O sistema deve permitir registra do Operário | Cadastrar Operário |

Fonte: Autor

**3.2.1. Requisitos Não Funcionais**

Tabela 4: Requisitos Não funcionais do sistema de gestão de Cutsheet

|  |  |  |
| --- | --- | --- |
| ***Nº*** | **Requisito** | **Descrição** |
| **RNF01** | Efectuar o Login | Para se ter acesso ao sistema deve ser com recurso a credenciais devidamente validados. |
| **RNF02** | Período de funcionamento | O sistema estará online 24h por dia |
| **RNF03** | N0 de Cutsheets por Recolha | Uma recolha só pose estar vinculada a no máximo três (3) cutsheets a decorrer |
| **RNF04** | A aprovacão da  requisicão | A requisição só é atendida no armazém depois da aprovação antes da supervisora e depois pelo coordenador da fábrica. |

Fonte: Autor

**1º Diagrama de Caso de Uso**

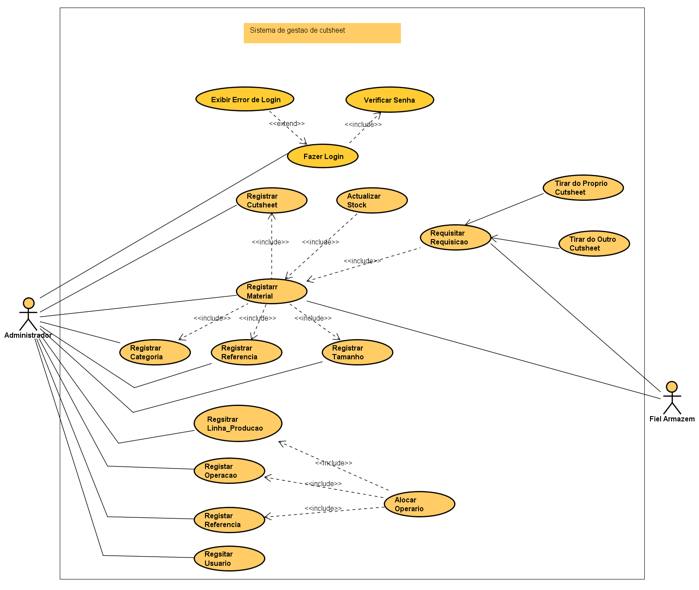


Figura 3: 1º Diagrama de Caso de Uso

Fonte: Autor

A figura acima é referente aos caso de uso que foi usado para modelar a primeira versão  
do sistema nele constam os actores, casos de usos e sua relação entre eles.

**Diagrama de Caso de Uso**

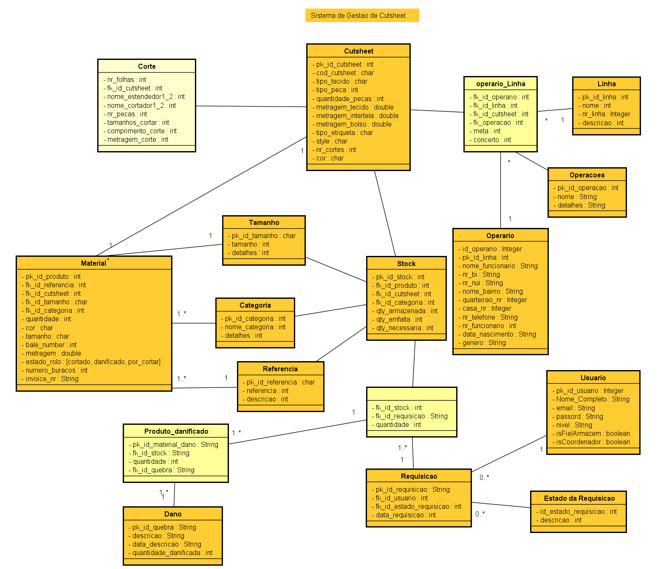


Figura 4: Diagrama de Caso de Uso

Fonte: Autor

# Referências bibliográficas

**Livros:**

Gil, A. C. Como elaborar projetos de pesquisa. São Paulo: Atlas, 1991.

GIL, A. C. Métodos e Técnicas de Pesquisa Social. São Paulo: Atlas, 6.ed, 2008.

MORESI, Metodologia da Pesquisa. 2003

BRAD, Hill. Pesquisa na internet. Rio de Janeiro: Campus, 1999

DEMO, Pedro. Avaliação qualitativa. São Paulo: Cortez, 1991.

Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta J. (2002) "Agile Software

Development Methods: Review and Analysis", Espoo 2002, VTT Publications 478.

PRESSMAN, R. S. Engenharia de Software. *São Paulo: Mc. Graw Hill*, *5 ed,* 2002.

PRESSMAN, Roger S. Engenharia de Software. *São Paulo: McGraw-Ill*, *6 ed*, 720, 2006.

Beck, K. and Fowler, M. (2000) "Planning Extreme Programming", Addison Wesley, 1a edição.  
Kruchten, P. (2000) "The Rational Unified Process – An Introduction", Addison-Wesley 2a edição.

Luiz, R. (2005) “Obtendo Qualidade de Software com o RUP”, TCC, Universidade de Uberaba.

SOMMERVILLE, I. Software Engineering. *Harlow, UK: Pearson Education*, *8a ed,* 2008.

SOMMERVILLE, I. *Engenharia De Software*. *9a ed,* 2011.

**Websites:**

http:www.cgnet. com.br/~walter/artigo.html. Acesso em: 20 jul. 2021

<https://www.luiztools.com.br/post/o-que-e-nodejs-e-outras-5-duvidas-fundamentais/>

<https://www.remessaonline.com.br/blog/metodologias-ageis/>

<https://www.devmedia.com.br/processo-de-teste-agil-x-tradicional/36854>

A MOZTEX.SA é uma fábrica pertencente a uma sociedade certificada pelo Ministério dos Recursos Naturais sob contracto de sociedade no ano de 2008 com o contrato assinado no quarto cartório Notória de Maputo na presença do notário Nassone Benbere (licenciado em direito), construída como uma sociedade anonima de responsabilidades limitadas entre Aga Khan Fund For Economic Development, S.A (AKFED), detentora de uma participação no valor de onze milhões de meticais, o acionista Alltex EPZ Limited, detentora de uma participação no valor de quatro milhões de meticais, e Nadir Badruddin Mohammad, detentor de uma participação de cinco milhões de meticais, sendo que o capital da sociedade integralmente subscrito em dinheiro é de vinte milhões de meticais. (Boletim da Republica – 24b de abril de 2008)

## 2.1.1. Objectivo MOZTEX.SA

1. Realização de toda ou qualquer actividade relativa á compra e venda ou á preparação, fiação, recelagem, cartatura, limpeza, engomadura, descoramento, tingimento, estampagem e acabamento ou manufactura por qualquer forma, com algodão, lã, seda, linho, fibra de planta, junta, canhado, seda artificial, fibra têxtil, fibra sintética ou alguma substancia similar; vender fio de algodão ou de outro produto manufaturado feito da mesma substancia ou de material similar e, tratar, utilizar e lidar com qualquer desperdício proveniente de quaisquer dessas operações, quer estes sejam resultado da actividade da sociedade ou outros, ou ainda dos produtos dos materiais; e a compra, venda, importação e exportação das substâncias mencionadas;
2. Manufaturar e processar todo tipo de vestuário e adornos, incluindo camisas, calcas, vestidos, malhas, roupas interiores, capas, chapéus e todo tipo de têxtil;
3. Realizar trabalho de costureiro, execução de robes e mantos, alfaiataria, fazendas de seda; fabricar e fornecer vestuário, lingerie e qualquer tipo de ornamento como coletes, peles, chapelaria de senhoras, luvas, passamanarias; serem representantes de tecidos, têxteis e de todo tipo de material, tiras/faixas, leques, perfumes e flores;
4. Realizar comercio geral e/ou empenhar-se no negocio de exportação, importação, intercambio de mercadorias, contratação, venda, compra; comercializar artigos de vestuários, tecidos, têxteis e mercadoria de toda classe e descrição por atacado ou retalho;
5. Assinar, comprar ou mesmo adquirir acções, quotas, obrigações, títulos de dívidas amortizáveis, e quaisquer títulos de quaisquer outras sociedades e investir e negociar com o dinheiro da sociedade de qualquer forma;
6. Realizar todo tipo de tarefas que directa ou indirectamente estejam relacionada com os pontos acima referidos;
7. Produção de vestuário.